Suppose I have a secret that I want to share across n parties such that any t subset can recover my secret, but any \( \leq t-1 \) subset learn nothing about my secret.

**Example:**

\[
\begin{array}{c}
\text{\( S_1, S_2, \ldots, S_n \)} \\
\downarrow \\
\text{\( \leq t-1 \) subset} \\
\downarrow \\
\text{Nothing}
\end{array}
\]

**Definition:** A \((t, n)\)-secret sharing scheme over a message space \( M \) and share space \( S \) is a tuple of efficient algs:

- **Share:** \( M \rightarrow S^n \)
- **Reconstruct:** \( S^t \rightarrow M \)

with the following properties:
Correctness: any \( t \) shares can be used to reconstruct \( m \):
\[
\forall m \in \mathcal{M}, (s_1, \ldots, s_n) \leftarrow \text{Share}(m) \\
\forall S \subseteq \{s_1, \ldots, s_n\} \text{ where } |S| = t \\
\text{Reconstruct}(S) = m
\]

Security: need at least \( t \) shares to learn anything about \( m \):
\[
\forall m_0, m_i \in \mathcal{M}, \forall I \subseteq \{1, \ldots, n\} \text{ where } |I| < t: \\
\text{Denote } (s'_1, \ldots, s'_n) \leftarrow \text{Share}(m_0), \\
(s'_1, \ldots, s'_n) \leftarrow \text{Share}(m_i)
\]
\[
\exists s_i \mid i \in I \approx \exists s'_i \mid i \in I
\]

today, these distributions will be identical.

Construction of \((n,n)\)-secret sharing:

For message space \( \mathcal{M} = \mathbb{F} \) and \( \mathcal{S} = \mathbb{F} \),

\[
\text{Share}(m): \text{Sample } r_1, \ldots, r_n \sim \mathbb{F} \\
\text{Define } r_i = m - \sum_{i=1}^{n} r_i \\
\text{Output } (r_1, \ldots, r_n)
\]

\[
\text{Reconstruct}(r_1, \ldots, r_n): \text{Output } m' = \sum_{i=1}^{n} r_i
\]

Correctness:
\[
\sum_{i=1}^{n} r_i = \sum_{i=1}^{n-1} r_i \left( m - \sum_{i=1}^{n-1} r_i \right) = m
\]

Security:
\( \forall m, m_i \in \mathcal{M} \), the \((n-1)\) share distributions are actually identical.
Shamir Secret Sharing: $(t,n)$-secret sharing scheme

For $M = \mathbb{F}$, $S = \mathbb{F}$ s.t. $1 < 1 > n$

Intuition: a polynomial of degree $t-1$ can be uniquely determined by $t$ points. For example, a line is defined by 2 points, and a parabola is defined by 3.

Linear Algebra Viewpoint: Given a point $x \in \mathbb{F}$ and a poly $f(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_{t-1} x^{t-1}$

We can view the evaluation $f(x)$ as an inner product:

$$f(x) = \langle (1, x, x^2, \ldots, x^{t-1}), (c_0, c_1, \ldots, c_{t-1}) \rangle$$

Thus, given $t$ distinct points, we can describe a linear system:

$$\begin{bmatrix}
1 & x_0 & x_0^2 & \cdots & x_0^{t-1} \\
1 & x_1 & x_1^2 & \cdots & x_1^{t-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_{t-1} & x_{t-1}^2 & \cdots & x_{t-1}^{t-1}
\end{bmatrix}
\begin{bmatrix}
c_0 \\
c_1 \\
\vdots \\
c_{t-1}
\end{bmatrix}
= 
\begin{bmatrix}
f(x_0) \\
f(x_1) \\
\vdots \\
f(x_{t-1})
\end{bmatrix}$$

Interpolating $f(x)$ is equivalent to solving the linear system above for the coeffs. For distinct $x_0, \ldots, x_{t-1}$, the Vandermonde matrix is invertible; thus, interpolating requires a matrix mul $V^{-1} \cdot \text{evals}$

Proof Sketch of Invertibility

If cols linearly dependent, then $\not\exists c_0, \ldots, c_{t-1}$ s.t.:

$$c_0 \begin{bmatrix}
| \\
| \\
| \\
| \\
| \\
\end{bmatrix} + c_1 \begin{bmatrix}
x_0 \\
x_0^2 \\
\vdots \\
x_0^{t-1}
\end{bmatrix} + \ldots + c_{t-1} \begin{bmatrix}
x_{t-1} \\
x_{t-1}^2 \\
\vdots \\
x_{t-1}^{t-1}
\end{bmatrix} = \begin{bmatrix}
0 \\
0 \\
\vdots \\
0
\end{bmatrix}$$
If this were true, then $x_0, \ldots, x_{t-1}$ are roots of a poly of deg $\leq t-1$ w/ coeffs $c_0, \ldots, c_{t-1}$. BUT a poly of deg $t-1$ has at most $t-1$ roots. Thus, $c_0, \ldots, c_{t-1} = 0$

**Construction**

Share($m$): Sample random coeffs $c_1, \ldots, c_{t-1} \in \mathbb{F}$

Define $f(x) := m + \sum_{i \leq t} c_i x_i$

Output $n$ points on $f$:

$\{s_i := (i, f(i)) \mid i \in [1, n]\}$

Reconstruct $((x_i, y_i), \forall i \in [t])$

- Interpolate the unique poly $f$ of deg $t-1$
  - Defined by those $t$ points
- Output $f(0)$

**Correctness:** Follows from the uniqueness of interpolation ($t$ points define a poly deg $\leq t-1$)

**Security:** Consider an arbitrary message $m \in \mathbb{F}$ and $I \subseteq [1, n]$ s.t. $|I| = t-1$. Define $\{x_i \mid \forall i \in [1, t-1]\} := I$. Consider arbitrary $y_i, \ldots, y_{t-1} \in \mathbb{F}$. What's the probability the shares for this subset are $(x_i, y_i), \ldots, (x_{t-1}, y_{t-1})$?

$$\Pr \left[V(0, x_i, \ldots, x_{t-1}) \begin{bmatrix} m \\ c_1 \\ \vdots \\ c_{t-1} \end{bmatrix} = \begin{bmatrix} m \\ y_1 \\ \vdots \\ y_{t-1} \end{bmatrix} \right] = \frac{1}{|\mathbb{F}|^{t-1}}$$

independent of $m$!

$a$ fixed $\hat{u} \in \mathbb{F}^n$
Another way to interpret: for any choice of \((t-1)\) shares and any message \(m\), there exists a unique poly \(f\) of deg \(t-1\) st.

\[ \forall i \in [1, t-1], f(x_i) = y_i \text{ and } f(0) = m \]

Thus, any \((t-1)\) shares can be consistent with the sharing of any message \(m\).

Now, we will describe a protocol for 2-PC (two-party MPC) for functions, expressible by Arithmetic Circuits.

We will show an elegant construction from secret sharing that targets semi-honest security, where we restrict corrupt parties to follow the protocol specification exactly but can try to extract info about the honest parties' input.

*there is an elegant way to make the protocol maliciously secure by using "info-theoretic MACs”

2-Party Computation for Arithmetic Circuits

1. Share Inputs
   - Alice \(X_A\): \(X_A, [X_A], [X_A], \leftarrow \text{Share}(X_A)\)
   - Bob \(X_B\): \(X_B, [X_B], [X_B], \leftarrow \text{Share}(X_B)\)

2. Compute on secret shares
   - \(X_A, [X_A], [X_A], \rightarrow\)
   - \(X_B, [X_B], [X_B], \leftarrow\)
3. Open final output shares

4. Reconstruct $[\text{Output}]_0 + [\text{Output}]_1 = \text{Output}$

**Protocol**

Idea: derive shares to intermediate wires incrementally

1. Both parties secret share their input elements with each other

2. For each addition gate in the circuit with inputs $[x], [y]$, the parties jointly derive shares to $[x+y]$ (the output share)

3. For each multiplication gate w/ inputs $[x], [y]$, parties jointly derive shares to $[x \cdot y]$

4. Once share of circuit outputs is derived, each party sends their share of the output
Computation on Secret Shares
Here we assume the (2, 2) scheme above \([x]_0 = r \in \mathbb{F}, \ [x]_1 = x - r\)

Adding Shares: 
\[
\begin{align*}
[x_A]_0 + [x_B]_0 &= [x_A + x_B]_0 \\
[x_A]_1 + [x_B]_1 &= [x_A + x_B]_1 \\
[x_A + x_B]_0 + [x_A + x_B]_1 &= [x_A]_0 + [x_B]_0 + [x_A]_1 + [x_B]_1 = x_A + x_B
\end{align*}
\]

Adding a constant \(c\): 
\[
\frac{c}{2} + [x_B] = [c + x_B]
\]

Multiplying by a constant \(c\): 
\[
c [x_B] = [cx_B]
\]

Multiplying Shares: will require setup + interaction

Beaver's Trick 91
Suppose parties already have secret shares of a random product: \([a], [b], [c]\) where \(a, b \in \mathbb{F}\) and \(c = ab\)

To multiply shares \([x]\) and \([y]\):
1. Locally compute shares to \([x - a]\) and \([y - b]\)
2. Send shares to jointly reconstruct \(E = x - a\) and \(\delta = y - b\); note that \(E\) and \(\delta\) are both one time pad encryptions of \(x\) and \(y\)
3. Locally compute shares \([z] = [c] + \delta [x] + E [y] - \frac{\epsilon \delta}{2}\)

Correctness
\[
\begin{align*}
[z]_0 + [z]_1 &= [c]_0 + \delta [x]_0 + E [y]_0 - \frac{\epsilon \delta}{2} \\
&= [c]_0 + \delta [x]_1 + E [y]_1 - \frac{\epsilon \delta}{2}
\end{align*}
\]
\[
\begin{align*}
C + \delta x &+ \varepsilon y - \varepsilon b \\
ab + (y - b)x + (x - a)y - \varepsilon b \\
ab + xy - bx + xy - ay - (x - a)(y - b) \\
ab + xy - bx + xy - ay - (xy - bx - ay + ab) \\
xy
\end{align*}
\]

Security: Information-theoretic!

How do parties obtain Beaver triples?

- Requires public-key cryptography or a trusted dealer
  \[\rightarrow\] Oblivious Transfer
  \[\rightarrow\] Garbled Circuits
  \[\rightarrow\] Somewhat homomorphic encryption

Need to generate one Beaver triple per multiplication gate (cannot reuse triples o/w break OTP)

2-PC in the Preprocessing Model

Preprocessing / offline stage: Parties generate \(M\) Beaver triples where \(M\) is an upper bound on multiplication gates. Note that this process is expensive, but independent of the future circuit/party input.

- Save these Beaver triples for use in the future; maybe waiting for data to be available or a computation to be agreed on
Online Stage: no expensive PK operations but requires communication linear in the # of mult. gates/inputs

- parties secret share inputs
- parties perform the 2-PC protocol using pregenerated triples (as long as # mult gates < M)
- parties reconstruct output